B.D.U.C.
BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 4

Hello again! Thanks and a quick mention for Dr. Andy Wright of BETASOFT in Birmingham for including a line about BDUC in the BETA BASIC newsletter. This resulted in enquiries from Spain, Sweden, Austria, Netherlands, Canada, India and Botswana.

C O N T R I B U T I O N S . C O N T R I B U T I O N S
Please send your contribution now! More material is required for forthcoming issues. Please help make this newsletter interesting and worthwhile. Anything related to Beta hardware or software.

Bernhard Lutz of Germany writes to say that he is in the army and was a member of the Beta user group set up by Per Henneberg Kristensen in Denmark. He was disapointed to receive only one newsletter and no reply to his letters after sending a large subscription fee, it now seems the club no longer exists. He sent this RAM Image splitter program which was developed for TRDOS version 4.11.

RAM IMAGE SPLITTER. BY BERNHARD LUTZ. TRDOS 4.11.

When using the "magic button" feature of the Beta disk interface the RAM image created on disk provides a useful method of saving programs quickly. However the block of code created is not really useful for any other purpose because of the way it has been saved. This program "strips" the screen from the magic file and enables the code to be saved as two blocks with or without the screen and enables use of tape and microdrive as well as disk.

The two code blocks created may then be loaded as required and may be transferred to microdrive or tape. Future enhancements to this program will be included soon. To use the RAM splitter program, type in listing 1 and save a security copy to tape. A formatted disk must be used to save the magic file on since the code expects to find the magic file on track 1 sector 0.

STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS..
A version of this program suitable for TRDOS 5.xx is now ready and will be included in the next issue.

The splitter program should be saved onto and loaded from another disk for the same reason as above using the name "boot".

How to use the RAM IMAGE splitter software.

1. With the software you wish to snapshot loaded and running, press the magic button at a suitable point as normal.
2. When the magic file has been saved. Load the splitter program "boot" from the second disk.
3. Option 7 may be used to test whether the copy will run correctly when it is loaded as a split version. If the test fails points A to E should be read and the splitter program run again.
4. Select option 8 (save & copy 16k/48k or screen$). A seven character file name should be entered, and the program will will split the previously saved RAM image as described. Files are saved in the following format. Screen$ are saved with "name"+"0". Programs are saved with "name"+"1" and "name"+"2".
5. A loader for the split image could be written as

```
10 CLEAR 27295
20 LET DOS=15363
30 RANDOMIZE USR DOS:REM:LOAD "name0"CODE     Optional line.
40 RANDOMIZE USR DOS:REM:LOAD "name1"CODE
50 CLS
60 RANDOMIZE USR DOS:REM:LOAD "name2"CODE
70 RANDOMIZE USR 16384
```

WHEN THE SPLIT RAM IMAGE FAILS TO WORK CORRECTLY.

A. Sometimes programs use Interrupt Mode 2, so it is necessary to use option 5 to alter the Interrupt Mode in the machine code part of the splitter program which is self loading from lines 9100 to 9130. This operates in a similar way to the "$" in a magic file name.
B. Some 16k programs test whether there is 48k available and self relocate so that a 16k program must sometimes saved as a 48k.
C. The disk may be full since the magic files are 192 sectors long.
D. The magic button may have been pressed at the wrong moment, if the program is testing what is on the screen, try using the magic button elswhere in the program.
E. Some programs should have any interrupts disabled using option 6, which modifies the self loading machine code.

The locations 27296 to 65535 may be poked as normal but locations 23296 to 27295 should be poked as (xxxxx-6873),YY.

)

Listing 1.

```
 10 CLEAR VAL "26999": LET dos=VAL "15363": LET im=VAL "86": LE
    T ei=VAL "251"
100 CLS : PRINT "RAM-IMAGE SPLITTER  1986 BL MENU"
110 PRINT '"1 SAVE SCREEN$ "''"2 GO TO DOS"''"3 SPLIT + COPY 48
    k"''"4 SPLIT + COPY 16k"''"5 INTERRUPT MODE: IM ";"1" AND i
    m= VAL "86";"2" AND im=VAL "94"''"6 INTERRUPT: ";"EI" AND e
    i=VAL "251";"DI" AND ei=VAL "0"''"7 TEST COPY";TAB VAL "16"
    ;"STOP GO TO BASIC"
120 LET A$=INKEY$: IF a$=" STOP " THEN RANDOMIZE USR VAL "0"
130 IF a$<"1" OR a$>"7" THEN GO TO VAL "120"
140 GO TO VAL a$*VAL "100"+VAL "100"
200 LET ix=VAL "32768": LET t=VAL "1": LET s=VAL "0": LET l=VAL
    "27": GO SUB VAL "1000"
210 GO SUB 4000: GO SUB 5000: LET ix=VAL "16384": LET n$=u$+"0"
    : LET l=VAL "6912": GO SUB VAL "1200"
220 RUN
300 RANDOMIZE USR VAL "15360"
310 RUN
400 LET mem=VAL "38240": GO TO VAL "510"
500 LET mem=VAL "5472"
510 GO SUB VAL "3000"
540 GO SUB VAL "2100": GO SUB VAL "2000": GO SUB VAL "4000": LE
    T ix=VAL "27296": LET l=mem:  LET n$=v$+"1": GO SUB VAL "12
    00"
550 LET ix=VAL "16384": LET l=VAL "4040": LET n$=v$+"2": GO SUB
    VAL "1200"
560 RUN
600 IF im=VAL "86" THEN LET im=VAL "94": GO TO VAL "100"
610 LET im=VAL "86": GO TO VAL "100"
700 IF ei=VAL "251" THEN LET ei=VAL "0": GO TO VAL "100"
710 LET ei=VAL "251": GO TO VAL "100"
800 CLS : LET l=VAL "22": PRINT "1 TEST 48k COPY "''"2 TEST 16k
    COPY "
805 PRINT ''"INTERRUPTMODE: IM ";"1" AND im=VAL "86";"2" AND im
    =VAL "94";''"INTERRUPT: ";"ENABLED" AND ei=VAL "251";"DISAB
    BLED" AND ei=VAL "0"
810 LET a$=INKEY$: IF a$<>"1" AND a$<>"2" THEN GO TO VAL "810"
820 IF a$="1" THEN LET l=VAL "150"
830 LET ix=VAL "27136": LET t=VAL "3": LET s=VAL "10": GO SUB
    VAL "1000"
840 GO SUB VAL "3000"
850 GO SUB VAL "2000"
860 GO SUB VAL "1600"
870 RANDOMIZE USR VAL "16384"
```

Listing 1 continued.

```
1000 GO SUB VAL "1500": POKE VAL "27005",INT (ix/VAL "256"): POK
     E VAL "27004",ix-VAL "256"*INT (ix/VAL "256"): POKE VAL "27
     007",t: POKE VAL "27009",s: POKE VAL "27011",l: RANDOMIZE U
     SR "27000": RETURN
1200 RANDOMIZE USR dos: REM: ERASE n$CODE
1210 RANDOMIZE USR dos: REM: SAVE n$CODE ix,l
1220 RETURN
1500 RESTORE VAL "9000": FOR a=VAL "27000" TO VAL "27044": READ
     b: POKE a,b: NEXT a: RETURN
1600 RESTORE VAL "9100": FOR a=VAL "16384" TO VAL "16422": READ
     b: POKE a,b: NEXT a: RETURN
2000 LET ix=VAL "16423": LET t=VAL "2": LET s=VAL "11": LET l=VA
     L "16": GO SUB VAL "1000": GO SUB VAL "1600": RETURN
2100 LET ix=VAL "27136": LET t=VAL "3": LET s=VAL "10": LET l=VA
     L "150": GO SUB VAL "1000": RETURN
3000 LET ix=VAL "22200": LET t=VAL "0": LET s=VAL "0": LET l=VAL
     "1": GO SUB VAL "1000"
3010 REM IF PEEK 22200=36 THEN LET im=94
3020 LET sp1=PEEK VAL "22209": LET sp2=PEEK VAL "22210"
3030 RETURN
4000 INPUT "ENTER NEW NAME: "; LINE v$
4010 IF LEN v$<1 OR LEN v$>7 THEN GO TO 4000
4020 INPUT "INSERT DESTINATION-DISK THEN PRESS ENTER: ";
4030 RETURN
5000 RESTORE VAL "9200": FOR a=VAL "40000" TO VAL "40011": READ
     b: POKE a,b: NEXT a: RANDOMIZE USR 40000: RETURN
8999 STOP : REM LOADER 27000,45
9000 DATA VAL "205",VAL "107",VAL "60",VAL "33",VAL "0",VAL "0",
     VAL "22",VAL "0",VAL "30",VAL "0"
9010 DATA VAL "62",VAL "1",VAL "6",VAL "1",VAL "14",VAL "0",VAL
     "245",VAL "197",VAL "213",VAL "205"
9020 DATA VAL "214",VAL "46",VAL "209",VAL "193",VAL "28",VAL "6
     2",VAL "16",VAL "187",VAL "204",VAL "161"
9030 DATA VAL "105",VAL "241",VAL "61",VAL "40",VAL "2",VAL "24"
     ,VAL "231",VAL "205",VAL "124",VAL "60"
9040 DATA VAL "201",VAL "30",VAL "0",VAL "20",VAL "201"
9099 REM RUNNER X,39
9100 DATA VAL "243",VAL "237",im,VAL "33",VAL "39",VAL "64",VAL
     VAL "17,VAL "0",VAL "91",VAL "1"
9110 DATA VAL "160",VAL "15",VAL "237",VAL "176",VAL "49",sp1,sp
     2,VAL "241",VAL "237",VAL "79"
9120 DATA VAL "241",VAL "237",VAL "71",VAL "241",VAL "225",VAL "
     209",VAL "193",VAL "217",VAL "8",VAL "253"
9130 DATA VAL "225",VAL "221",VAL "225",VAL "225",VAL "209",VAL
     "193",VAL "241",ei,VAL "201"
9199 REM LDIR 32768-16384
9200 DATA 33,0,128,17,0,64,1,0,27,237,176,201
```

MACHINE CODE AND BETA DOS. BY HENDRICK BROOTHAERS.
FOR 4.XX DOS.

Here is another interesting article from Hendrick Broothaers in Belgium. Part one is in this issue and the second part will follow in issue 5. The first part contains an explanation of useful Beta DOS routines and part two has examples of how to use them from machine code.

These routines are only for version 4.xx through 5.xx. All the routines can be accessed with a unique CALL address. The CALL address is 3BFD (15357). When this address is CALLed the number of the requested routine must be in the C register. The contents of some other registers and some DOS variables provide parameters and control over the routines.

In order to gain access to these routines the DOS must be switched on, this is done by a CALL 3CØ6 (15366) , followed by a PUSH HL. (the PUSH HL puts a RETURN address on the stack which will automatically switch the DOS off when we leave our code via a RETurn) The next thing to do is to set the necessary registers (if this was not done before), load the C register with the desired routine number and do a CALL 3BFD (15357) followed by a RETurn.
Here is an explanation of the DOS variables used by the routines.

5CD1/2 (23761/2)	Auto run line number. Used when a BASIC file is written to disk using routine 12.
5CD2 (23762)	Must hold the array character when loading a DATA file (character or number array) with routine 14.
5CD7 (23767)	Buffer address for a read or write of one sector.
5CD8 (23768)	Used in routine 14 if 5D1Ø is different from zero.
5CDD (23773) to	Filename. (8 characters).
5CE4 (2378Ø)	
5CE5 (23781)	Filetype (B) BASIC (C) CODE (D) DATA.
5CE6 (23782)	Total length for BASIC.
5CE7 (23783)	Load address for CODE or DATA.
5CE8/9 (23784/5)	Program length for BASIC or Byte length for CODE or DATA.
5CEA (23786)	File length in sectors.
5CEB/C (23787/8)	Sector and track where file starts on disk.
5DØF (23823)	Number a file has in the catalogue, (used by routine 1Ø).
5D1Ø (23824)	Controls subroutine 14 (see text).

)

DESCRIPTION OF THE ROUTINES.
The routines are numbered 0 to 20 (or in HEX 00 to 14).

ROUTINE 0 (# 00)

The currently selected drive is restored to track 0 and the break key is checked.

ROUTINE 1 (# 01)

Selects the drive who's number is in the A register.

ROUTINE 2 (# 02)

Positions the head on the track number that is in the A register on routine entry.

ROUTINE 3 (# 03)

Store A register in 5CFF (23807) (= sector for read or write).

ROUTINE 4 (# 04)

Store HL register in 5D00 (23808) (= buffer address).
Note: (routines 3 and 4 have no specific stand-alone use, they are called from within other routines).

ROUTINE 5 (# 05)

READ from disk. Any sector/track and number of sectors:

```
Registers on entry: B = number of sectors to read
                    DE = track/sector to read from
                    HL = buffer address for read data
```

ROUTINE 6 (# 06)

WRITE to disk. Any sector/track and number of sectors.

```
Registers on entry: B = number of sectors to write
                    DE = track/sector to write to.
                    HL = buffer address for write data
```

ROUTINE 7 (# 07)

CATalogue to stream whose number is in the A register.

ROUTINE 8 (# 0 8)

Read file info from track zero to DOS variables 5CDD to 5CEC (23773 to 23788). For file who's number is in the A register. (16 bytes are moved).

ROUTINE 9 (# 09)

Write file info from DOS variables 5CDD to 5CEC (23773 to 23788) to disk. For file who's number is in the A register. (16 bytes are moved).

)

ROUTINE 10 (# 0A)

Search disk catalogue for file who's name and type are in the DOS variables 5CDD to 5CE5 (23773 to 23781). On completion address 5D0F has the number the file has in the catalogue or FF if the file is not found in the catalogue.

ROUTINE 11 (# 0B)

SAVE a non-basic file. File name and type must be in DOS variables 5CDD to 5CE5 (23773 to 23781), DE = length in bytes HL= start address.

ROUTINE 12 (# 0C)

SAVE a BASIC file. On entry: File name and type must be in DOS variables 5CDD to 5CE5 (23773 to 23781). Memory 5CD1 (23761) (LO)-5CD2 (23762) (HI) must have the autorun line number.

ROUTINES 13-15-16-17 (# 0D-0F-10-11)

These routines are one and the same. They are used from within DOS as an exit from all the other routines.

ROUTINE 14 (# 0E) (explained in detail later).

LOAD (BASIC-CODE-DATA). The operation depends on the contents of locations 5CD2 (23762),5CD7 (23767),5D10 (23824) and the registers A'- HL - DE. The file name and type must be in DOS variables 5CDD to 5CE5 (23773 to 23780).

ROUTINE 18 (# 12)

ERASE the file who's name and type are in variables 5CDD to 5CE5 (23773 to 23780).

ROUTINE 19 (# 1 3)

LDIR memory to DOS variables 5CDD to 5CEC (23773 to 23788) HL is memory pointer.

ROUTINE 20 (# 14)

LDIR DOS variables 5CDD to 5CEC (23773 to 23788) to memory. HL is memory pointer.

Details on routine 14 (# 0E)

Routine 14 is the most complex routine,it is used to LOAD a file I will explain the different cases. You must ALWAYS make sure that the file name and type are in DOS variables 5CDD to 5CE5 (23773 to 23780) before CALLing routine 14, otherwise a "NO FILE" message is generated and the operation is aborted.

For a BASIC file:

5D10 = 00   A register = 00   then CALL routine 14
-there must be enough room below RAMTOP
-any previous BASIC is wiped out

For a DATA file (number or character array):

5D10 = 00   A register = 00   HL register = 0000
5CD2 = array name expl. 81 for number array "a"
C1 for character array "a$"
the routine DIMensions the array before LOADING the data.

For a CODE file:

5D10 has following controls over routine 14 :
5D10 = 00 = LOAD a complete file
5D10 = FF = LOAD one sector only
5D10 different from 00 and from FF = write one sector to disk.

-with 5D10 = 00

-A register = 00 LOAD the file to the address it was SAVED from
-A register = 03 LOAD the first sectors of a file to the address in HL register. D register = number of sectors to load
-A reg. not 00 and not 03 = LOAD the file to the address in HL

-with 5D10 = FF

LOAD the sector who's number is in the L register to the address specified in loc's 5CD7-5CD8.
example: if L = 5, the fifth sector of the file is LOADED.

-with 5D10 different from 00 and from FF

WRITE one sector to a file. As above the sector number must be in the L register and the memory address in loc's 5CD7-5CD8 before routine 14 is CALLed.

This concludes part one of this article, the next BDUC newsletter will have the second part with examples of how to use the routines from machine code.

B D U C, 2, DOWNHAM AVENUE, RAWTENSTALL, ROSSENDALE,
LANCASHIRE, BB4 8JY. ENGLAND. PHONE:010 44 706 218354